LArIAT SAM Metadata Description

J. St. John, R. Johnson

October 22, 2015

Contents

1 Acknowledgements

This document was liberally plagiarized from the MicroBooNE SAM Metadata Proposal [1]. We hereby acknowledge the work of B. Carls, E. Church, H. Greenlee, M. Kirby, Z. Pavlovic, and S. Wolbers.

2 SAM Overview

SAM is primarily a file catalog of files belonging to a particular experiment. The file catalog is implemented as a physical database somewhere. Users do not (usually) interact with the database directly, but rather client programs send requests to an http(s) server called a SAMweb server. The SAMweb server responds to client requests, which responses may include returning information extracted from the database. The SAMweb server can also update the SAM database.

The SAMweb server interface supports its own query language (not SQL) for identifying collections of files. Users are allowed to record and name queries permanently in the SAM database. Such a memorized query is called a *dataset definition*. A SAMweb server can be requested to execute a query (whether memorized in the form of a dataset definition or not) and return a list of files. A SAMweb server can likewise be requested to memorize the list of files obtained as a result of a query, which list is called a *snapshot*. Finally, a snapshot can be used to define a *project*, which consists of a set of files that will be scheduled for delivery to a set of worker jobs.

For LArIAT, we expect to take data in a large variety of run configurations. The database should have enough fields to reflect the variety of these configurations so that when an analyzer wishes to analyze "all the files" of a specific type, those files should be able to be defined by SAM fields. It is the purpose of this note to define the database fields that can be queried. While the definition of what is in those database fields may change throughout LArIAT running and analysis, the number of fields probably will not. It is difficult to go back and increase the number of fields in a database. For this reason, we have added ten fields at the end (called option.0-9) that can be used for later expansion.

This database is expected to index all of the files in the analysis stream, from raw data through final TTrees. Therefore there are fields for parent files and daughter files within the database. These allow a record of the history of an analysis.

2.1 SAM Resources

There are several Fermilab redmine sites that contain useful information about SAM, and related products. The Fermilab redmine home page is

`https://cdcvs.fnal.gov/redmine`

A full listing or Fermilab redmine projects and subprojects can be found at

`https://cdcvs.fnal.gov/redmine/projects`.

Specific project redmine pages have urls like

`https://cdcvs.fnal.gov/redmine/projects/`*project-name*,

where *project-name* is the name of a redmine project or subproject. Some redmine projects that are relevant for SAM are as follows.

- `sam-main` - Main SAM project page.

- `sam-web` - Contains information about metadata and SAM query language.
- `sam-web-client` - Command line and python clients.
- `filetransferservice` - File transfer service.
- `ifdhc` - Main page for data handling tools.
- `ifdh-art` - Art data-handling interface (art SAM client).

Note that file transfer service and data handling tools are not part of SAM per se, but they do interact with SAM. They are also outside the scope of this document. They are listed here for informational purposes.

3 SAM Metadata

SAM associates various kinds of information with files, which information collectively is called SAM metadata. The primary purpose of SAM metadata is to be used in constructing queries that are part of dataset definitions, and secondarily as a repository for information that may be of interest for other purposes.

3.1 SAM Database

The SAM database is designed to have a fixed schema that is the same for every experiment (different experiments will have their own database instances). The SAM design does include a provision for adding experiment-specific metadata (see Sec. 3.3), but the underlying database schema doesn't change. The SAM database is not a good substitute for having experiment-specific databases with schemas defined for particular experiment-specific purposes (such as trigger database, runs database, Monte Carlo database, etc.).

3.2 Predefined Metadata Fields

A list of predefined metadata fields can be found in Robert Illingworth's metadata workshop talk [2], as well as in the `sam-web` redmine wiki [3]. A list of the most relevant predefined metadata fields is reproduced in Table 1.

In the following sections, we will give further explanations and suggestions regarding some of the fields listed in Table 1.

3.2.1 File Names and File Ids.

Files in the SAM database are uniquely identified by either a file id. (an integer) or a file name (a string). In database terms, the file id. is the primary key, and the file name is a unique key. The file id. is generated and used internally by SAM. The file name is assigned by the user and can be anything.

Lariat data files are named either:

Table 1: Predefined metadata fields.

Name	Type	Required?	Predefined Values?	Description
file_id	integer	yes	no	File id.
file_name	string	yes	no	File name
file_size	integer	yes	no	File size in bytes
file_type	string	yes	yes	File type
file_format	string	no	yes	File format
data_tier	string	no	yes	Data tier
group	string	no	yes	Group
crc	struct	no		File checksum
crc.crc_value	integer	no	no	Checksum value
crc.crc_type	string	no	no	Checksum type
application	struct	no		Application
application.family	string		no	Application family
application.name	string		no	Application name
application.version	string		no	Application version
parents	struct array	no	no	List of parent files
parent.file_name	string	no	no	Parent file name
parent.file_id	string	no	no	Parent file id.
event_count	integer	no	no	Number of events
first_event	integer	no	no	First event number
last_event	integer	no	no	Last event number
start_time	date	no	no	File start time
end_time	date	no	no	File end time
data_stream	string	no	yes	Stream
runs	struct array	no		List of runs
run.run_number	integer		no	Run number
run.subrun_number	integer		no	Subrun number
run.run_type	string		yes	Run type

lariat_ physics_ *run_ number_ spill_ number*.dat
lariat_ r*run_ number_* sr*spill_ number_ date/time*.root

The former is a raw data file and the latter is an artdaq file. The run number and the spill number in the raw file are just that with no leading zeros. The run number in the artdaq file is six digits with leading zeros. The spill number is four digits with leading zeros. Early in the run the spill number was two digits.

3.2.2 Data Organization

A Lariat run is a number of spills with the same beam and trigger configuration. Each spill is written to disk as a single file. That file will be registered with SAM as a subrun and transfered to dCache.

3.2.3 File Type, File Format, Data Tier

The file format describes the physical format of a file. The file type is used to give a higher level description of what the file is intended to be used for. The data tier represents the stage in a typical processing chain.

Some suggested values for the file data are listed in Table 2. For DAQ data all files are typed "data," file format is "raw" for ".dat" files and "artroot" for ".root" files, and data tier is "raw" for all files.

3.2.4 Group

The group field is intended to stand for a group within an experiment, rather than the whole experiment. For example, the group field could be used by a particular physics group to label its particular Monte Carlo files. However, initially it will be sufficient to have one catch-all group ("lariat") for the whole experiment (see Table 2).

3.2.5 File Size and CRC

File size is the file size of the file on disk. The check-sum is computed by SAM later and is not entered before the initial catalog.

3.2.6 Application Information

The program that created the file is identified in SAM metadata by a 3-tuple of strings which represents the application (family, name, version). SAM does not require these fields to be predefined in the database, so they can be anything the user chooses.

In the case of art framework programs, the application family will always be "`art`." Likewise, art framework programs will usually use the general purpose larsoft executable "`lar`." In this case, rather than storing the application name as simply "`lar`," which doesn't convey much new information, we will store the application name as the process name parameter stored in the fcl job file, or specified on the command line using option "`--process-name`." Of course, there may be special

Table 2: Suggested values for some enumerated metadata fields.

Field	Values
file_type	data (all files) mc unknown
file_format	raw (.dat files) artroot (.root files) root tar unknown
data_tier	generated-beam-only generated-full simulated raw (all files) sliced reconstructed root-tuple root-histogram unknown
group	lariat
crc_type crc_value	
data_stream (not used)	all beam-only cosmic_ray random
run_number	DAQ run number
subrun_number	spill number within a run
run_type	physics pedestal channel_scan test

cases (including the daq system) where some executable other than `lar` is used. In these cases, the application name should reflect the name of the actual executable.

In the case of art programs, the combination of the application name and the data tier will give a general idea of the purpose and configuration of the program, but not full details. We will definitely want to have more information in the metadata about how art programs were configured via their job fcl files. Our proposal is to add this additional information using experiment-specific metadata (see Sec. 3.3.3).

Application information is not set for DAQ files.

3.2.7 Parentage Information

The SAM metadata contains a field for identifying the parent files (of which there can be more than one) for each file. It was the intention of the SAM designers that this field should be used to store the immediate parents, and we propose to do that.

Parentage information is not set for DAQ files.

3.2.8 Data Stream

As its name implies, this metadata field is intended to identify different output streams that are produced by the same program. DAQ data has only one data and is unused for these files.

3.2.9 Run Information

In the SAM metadata, runs are identified by a 3-tuple consisting of a run number (an integer), a sub-run number (integer) and a run type (a string). For data, the run number, sub-run number, and run type will normally be assigned by the daq system and carried forward to all descendant files. Some proposed run types are listed in Table 2. Files can have multiple runs. The runs "struct array" should be filled accordingly.

For SAM calls, the run number and subrun number are coded as "run_number.subrun_number" and called as "run_number".

3.3 Experiment-Specific Metadata Fields

SAM metadata are designed to be extensible by the addition of experiment-specific metadata fields, which are called parameters in SAM documentation. These experiment-specific metadata, or parameters, are represented in metadata definitions and queries as key-value pairs of the form *category.name = value*, where *category*, *name* and *value* are whatever the experiment chooses them to be (other than predefined categories and names, such as those listed in Table 1). Only scalar data types are supported for parameter values (no structs or arrays).

Table 3: Beam parameters.

Parameter	Type	Values	
secondary.momentum	float	(MeV/c)	IFBEAM
secondary.polarity	string	Positive/Negative	IFBEAM
secondary.intensity	float	SC1 counts	IFBEAM
tertiary.magnet_current	float		IFBEAM
tertiary.magnet_polarity	string	Positive/Negative/Unknown	IFBEAM
tertiary.USTOF	string	On/Off/Partial/Unknown	IFBEAM
tertiary.DSTOF	string	On/Off/Partial/Unknown	IFBEAM
tertiary.MWPC1	string	On/Off/Unknown	IFBEAM
tertiary.MWPC2	string	On/Off/Unknown	IFBEAM
tertiary.MWPC3	string	On/Off/Unknown	IFBEAM
tertiary.MWPC4	string	On/Off/Unknown	IFBEAM
tertiary.cherenkov1	string	On/Off/Partial/Unknown	IFBEAM
tertiary.cherenkov2	string	On/Off/Partial/Unknown	IFBEAM
tertiary.beam_counters	string	On/Off/Partial/Unknown	IFBEAM
tertiary.halo_paddle	string	On/Off/Partial/Unknown	IFBEAM
tertiary.cosmic_counters	string	On/Off/Unknown	Acciarri
tertiary.muon_range_stack	string	On/Off/Partial/Unknown	IFBEAM
tertiary.number_MuRS	integer	0-16	IFBEAM
tertiary.punch_through	string	On/Off/Partial/Unknown	IFBEAM

3.3.1 Beam Parameters

Refer to Table 3 for proposed names and values of beam parameters. The secondary beam parameters for the beam incident on the secondary target are prefaced with "secondary" and the beam parameters after the secondary target are prefaced with "tertiary".

3.3.2 Detector Parameters

The proposed detector status parameters are listed in Table 4.

3.3.3 FCL Configuration

In addition to the application name and version, art programs are fully specified by their fcl configuration. Since the fcl configuration is much too complex to be fully captured by SAM metadata, we propose to treat the fcl configuration similarly as the trigger configuration, by specifying the fcl name and version, which name and version can be used to reference an external fcl database (see Table 5).

FCL Configuration is not used or set for DAQ data.

Table 4: Detector parameters.

Parameter	Type	Values	Source
detector.cathode_voltage	float	-999999.99 (error)	IFBEAM???
detector.shield_voltage	float	-999999.99 (error)	IFBEAM
detector.induction_voltage	float	-999999.99 (error)	IFBEAM
detector.collection_voltage	float	-999999.99 (error)	IFBEAM
detector.pmt_ham	string	On/Off/Unknown	IFBEAM
detector.pmt_etl	string	On/Off/Unknown	IFBEAM
detector.sipm_ham	string	On/Off/Unknown	IFBEAM
detector.sipm_sensl	string	On/Off/Unknown	IFBEAM

Table 5: Fcl parameters.

Parameter	Type
fcl.name	string
fcl.version	string

3.3.4 Project Configuration

For processed data files it will be necessary to record not just the configuration of the most recent program that produced the file (the fcl configuration), but the configuration of an entire processing chain. Typical examples of this are MC files, where the generator configuration is relevant at later processing stages. Therefore we propose to include a project parameter (see Table 6).

Project Configuration is not used or set for DAQ data.

3.3.5 Offline Streaming and Filtering

At some point, we will certainly want to create sparse datasets for various specialized purposes, like particular physics analyses. For many purposes, the combination of application and fcl configuration will be adequate to specify how such filtering was done. To handle, in addition, the case where the same filtering program (or any

Table 6: Project parameters.

Parameter	Type
lariat_project.name	string
lariat_project stage	string
lariat_project.version	string

Table 7: Filter parameters.

Parameter	Type
`filter.name`	string

program) produces multiple output files, we propose to define a filter name parameter (see Table 7).

Offline Streaming and Filtering is not used or set for DAQ data.

4 Revision History

- Nov. 22, 2015: Revised to reflect Run 1 parameters (RAJ).

References

[1] E. Church, *et al.*, MicroBooNE DocDB #2414, `http://microboone-docdb.fnal.gov:8080/cgi-bin/RetrieveFile?docid=2414&filename=metadata.pdf&version=4`

[2] R. Illingworth, `https://cdcvs.fnal.gov/redmine/documents/594`.

[3] `https://cdcvs.fnal.gov/redmine/projects/sam-web/wiki/Metadata_format`.